NPS52-86-003

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A SINGLE BOARD MULTIPROCESSOR FOR REAL-TIME CONTOUR SURFACE DISPLAY GENERATION

Michael J. Zyda

Robert A. Walker

January 1986

Prepared for:

ef of Naval Research
ington, VA 22217

51-86-003

NAVAL POSTGRADUATE SCHOOL
Monterey, California

The work reported herein was supported in part by the Foundation Research Program of the Naval Postgraduate School with funds provided by the Chief of Naval Research.

This report was prepared by:

[illegible]

MICHAEL J. ZYDA
Professor of
Computer Science

Reviewed by:

[illegible]

VINCENT Y. LUM
Chairman
Department of Computer Science

Released by:

[illegible]

KNEALE T. MARSHALL
Dean of Information and
Policy Science

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS52-86-003	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Single Board Multiprocessor for Real-Time Contour Surface Display Generation		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael J. Zyda Robert A. Walker		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61152N; RR000-01-NP N0001485WR41005
11. CONTROLLING OFFICE NAME AND ADDRESS Chief of Naval Research Arlington, VA 22217		12. REPORT DATE January 1986 13. NUMBER OF PAGES 21
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) contouring, contouring tree, contour surface display generation, real-time display generation		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We present in this study an integrated systems design for a VLSI multiprocessor capable of generating contour surface displays in real-time (one-thirtieth of a second). We begin by examining an application that requires real-time contour surface display generation [10]. We sketch some outlines for an architecture based upon a decomposable algorithm recently published [12]. We then propose an architecture for a single board VLSI contour surface display generator that is pluggable into the Multibus of the Silicon Graphics, Inc. IRIS workstation.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A Single Board Multiprocessor for Real-Time Contour Surface Display Generation ‡

Michael J. Zyda and Robert A. Walker

Naval Postgraduate School,

Code 52, Dept. of Computer Science,

Monterey, California 93943-5100

ABSTRACT

We present in this study an integrated systems design for a VLSI multiprocessor capable of generating contour surface displays in real-time (one-thirtieth of a second). We begin by examining an application that requires real-time contour surface display generation [10]. We sketch some outlines for an architecture based upon a decomposable algorithm recently published [12]. We then propose an architecture for a single board VLSI contour surface display generator that is pluggable into the Multibus of the Silicon Graphics, Inc. IRIS workstation.

Categories and Subject Descriptors: I.3.1 [**Hardware Architecture**]: architectures, parallel processing, VLSI implementations; I.3.2 [**Graphics Systems**]: multiprocessing systems; I.3.5 [**Computational Geometry and Object Modeling**]: data structures, discrete planar contours, modeling molecules, surface approximation, surface generation, surface representation, surfaces, 3D graphics; I.3.6 [**Methodology and Techniques**]: contouring, interactive systems, parallel processing; I.3.7 [**Three-Dimensional Graphics and Realism**]: line drawings, line generation algorithms, real-time graphics, surface plotting, surface visualization, surfaces; I.3.m [**Miscellaneous**]: VLSI;

General Terms: Algorithms, architecture;

Additional Key Words and Phrases: contouring, contouring tree, contour surface display generation, real-time display generation;

1. Introduction

Contour surface display generation is one of the most frequently used applications graphics algorithms [1, 3, 8-16]. A contour surface display is a visual representation of a surface by the

‡ This work has been supported by the NPS Foundation Research Program.

collection of lines formed when that surface is intersected by a set of parallel planes (Figure 1). The lines formed on each of those planes are called contours. A contour represents the set of points that belong to both the surface and the particular intersecting plane. Contour surface displays are used in X-ray crystallography, computer-aided tomography, and other applications for which grid data is collected. Contour surface display generation is generally depicted as a computationally slow operation whose output is sent to a plotter or film recorder. One publication has described an architecture, and produced a feasibility determination for a VLSI based contour surface display generator [10]. The architecture proposed in that study, while appropriate for its level, does not provide enough detail for actual implementation. Neither does that study consider some of the issues that occur when one actually designs a working system. This study attempts to remedy those deficiencies by providing an integrated systems architecture for a real-time contour surface display generator that is both precise in detail, and technologically realistic.

1.1. Contour Surface Display Generation is Slow

Our initial premise for this study is that contour surface display generation on a single processor system, such as a graphics workstation with a floating point accelerator, is too slow to be of any use for interactive applications requiring such a display. This premise is based upon [10], and is reinforced by statements found in the literature. A number of papers have been written documenting "breakthroughs" that increase the speed of contour surface display generation. One author has reported that his contour surface display generation subroutine used one second of central processor time on NCAR's Control Data 7600 [9]. Although a contour surface display generation program of this speed is useful for static situations, it is unacceptable for applications that generate a succession of contour surface displays in response to contour level changes read from a control dial. Such a program requires that a new contour surface display be generated, distributed, and displayed in real-time, typically one-thirtieth of a second for current display technology [6].

One application in which real-time contour surface display generation is important is the determination of molecular structures from the electron density data generated by X-ray

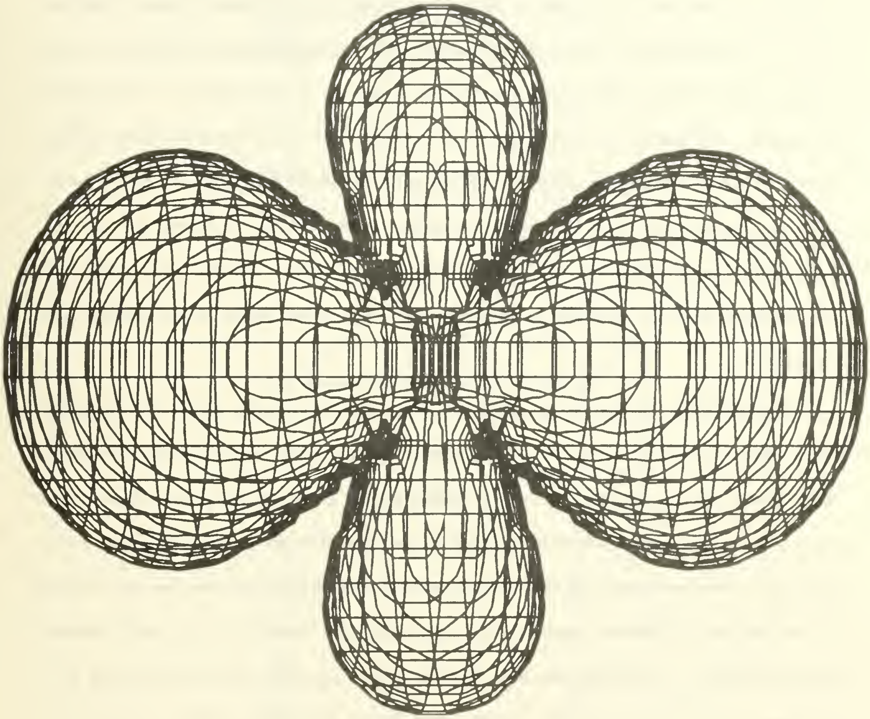
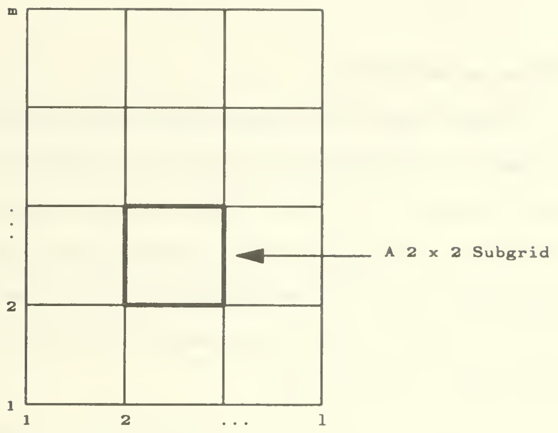


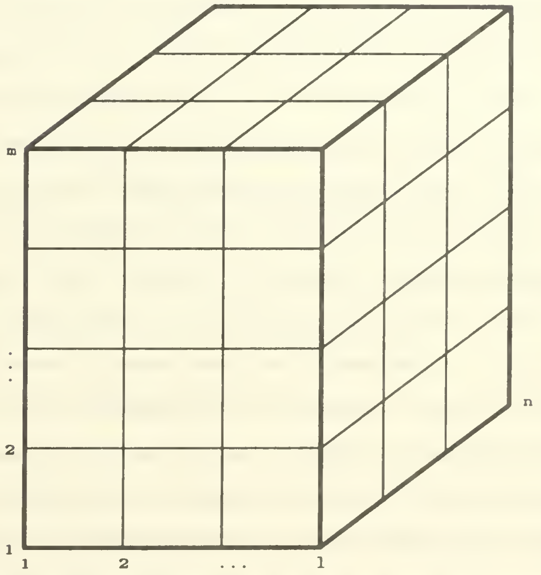
Figure 1
Contour Surface Display Generated from a Hydrogen Atom Wavefunction Squared (3dz2 orbital)

crystallography [1]. Such an operation is executed interactively by using a computer graphics program that displays a Dreiding (stick) model of the molecule inside a contour surface display of the corresponding region of the molecule's electron density grid. In addition to the graphics function, the computer program monitors a series of signals generated by the user, while the user is turning the various knobs on a control console [17]. The values read from these knobs are interpreted by the program as modifications to either the molecule or the surface display. Modifications to the molecule take the form of bond rotations or bond lengthenings. Modifications to the contour surface display take the form of an increase or decrease of the contour level. The goal of this process is to produce the stick model of the molecule that best fits inside the given electron density data set. The user can determine whether or not the model fits the density grid by modifying the contour level. shrinking the contour surface to the molecule. Similarly, the user can expand the contour surface from the stick model for better visibility. This function requires that the hardware have the capability to rapidly change the contour display as its contour level changes.

We know from [13] that the generation of a contour surface display, such as those required by the above application, cannot be accomplished in real-time using a conventional uniprocessor. This failure is due to the fact that contour surface display generation algorithms require many more instructions executed per second than can be provided by currently available uniprocessors. In the past, this limitation of the conventional processor has relegated such applications to either the non real-time environment (waiting a few minutes for each display), or to the equally unsatisfying environment of motion picture film. Because of this, the author of [10] looked for a multiprocessor solution to the real-time contour surface display generation problem. At the time of that study, efficient multiprocessor solutions meant VLSI solutions. Consequently, the multiprocessor architecture proposed in that study was qualified by the need for implementation in VLSI. This study continues with that qualification.



A 2D Grid of Size l x m Has
(l - 1) x (m - 1) 2 x 2 Subgrids.



A 3D Grid of Size l x m x n Has
l x (m - 1) x (n - 1) + m x (l - 1) x (n - 1) + n x (l - 1) x (m - 1)
2 x 2 Subgrids

Figure 2
2 x 2 Subgrid Count for 2D and 3D Grids

1.2. Contouring Definitions

A contour surface is a visual display that represents all points in a particular region of three-space $<x,y,z>$ which satisfy the relation $f(<x,y,z>)=k$, where k is a constant known as the contour level (Figure 1). The function f represents a physical quantity which is defined over the three-dimensional volume of interest. The visual display created by this algorithm is the collection of lines that belong to the intersection of both the set of points that satisfy the relation $f(<x,y,z>)=k$, and a set of regularly spaced parallel planes that pass through the region of three-space for which the relation is defined.

For this study, the function f is approximated by a discrete, three-dimensional grid created by sampling that function over the volume of interest. The three-dimensional grid contains a value at each of its defined points that corresponds to the physical quantity obtained from the function, i.e. the value associated with point (x_0,y_0,z_0) is v_0, where $f(x_0,y_0,z_0)=v_0$.

A decomposable algorithm for contour surface display generation is described in [12]. That algorithm is constructed from a two-dimensional contouring algorithm that is used to contour all the possible planar, orthogonal, two-dimensional grids of a larger three-dimensional grid. The two-dimensional contouring algorithm of that study is comprised of components, called algorithm components, that operate on individual 2 x 2 subgrids of a larger two-dimensional grid. (Note: a 2 x 2 subgrid is defined to be that portion of the two-dimensional grid bounded by four adjacent grid points.) In the algorithm, the computations necessary for generating the contour lines for a single 2 x 2 subgrid are independent from those required for any other 2 x 2 subgrid. If we compute the contours corresponding to contour level k for all 2 x 2 subgrids of a two-dimensional grid, then we will have determined the complete set of contours for that grid. If we compute the contours corresponding to contour level k for all possible 2 x 2 subgrids of the larger three-dimensional grid, then we will have the complete contour surface display for that grid. The assemblage of the contours created by this process, i.e. the simultaneous display of all the contours created for all 2 x 2 subgrids of the larger three-dimensional grid, produces a "chicken-wire-like" contour surface display (Figure 1). The full development of this algorithm can be found in

[11-13]. We refer to the results of those studies, and consequently, do not cover the algorithm here in great detail. We only note that for the largest three-dimensional grid of interest for the above application, a 30 x 30 x 30 grid, this means the potential for 75,690 parallel operations (see Figure 2 and [1]).

2. Architectural Goals for the Contour Surface Display Generator

The first goal in the design of the contour surface display generator is to build a system that meets the performance requirements, i.e. a new contour surface display computed from a 30 x 30 x 30 grid, and delivered to a display device in one-thirtieth of a second. This is an ambitious goal but it must be noted that one-thirtieth of a second is the maximum amount of time allowable for the operation. Any longer amount of time does not provide the viewer smooth transitions between successive contour surface displays. This goal says nothing about the load time of the 30 x 30 x 30 grid to the special piece of hardware that computes the contour surface display. Consequently, we allow solutions that pre-load the grid.

The second goal for the construction of the contour surface display generator is that we be able to plug it into an existing graphics system with minimal hardware and software changes. For the purposes of this study, the target graphics system is chosen to be the Silicon Graphics, Inc. IRIS workstation (see Figure 3 and [2]). The Silicon Graphics, Inc. IRIS is currently the highest performance graphics system that best matches the selected application's goals.

3. Architecture of the Contour Surface Display Generator

There is not enough space in this paper to present the complete architecture of the contour surface display generator. The reader is instead referred to [8]. An overview architectural description is that the contour surface display generator is comprised of four subsystems: (1) the array of algorithm component processors, (2) the controller for that array of processors, (3) the algorithm component processor itself, and (4) the interface to the graphics system. Figure 4 shows how the four subsystems relate to the target graphics system.

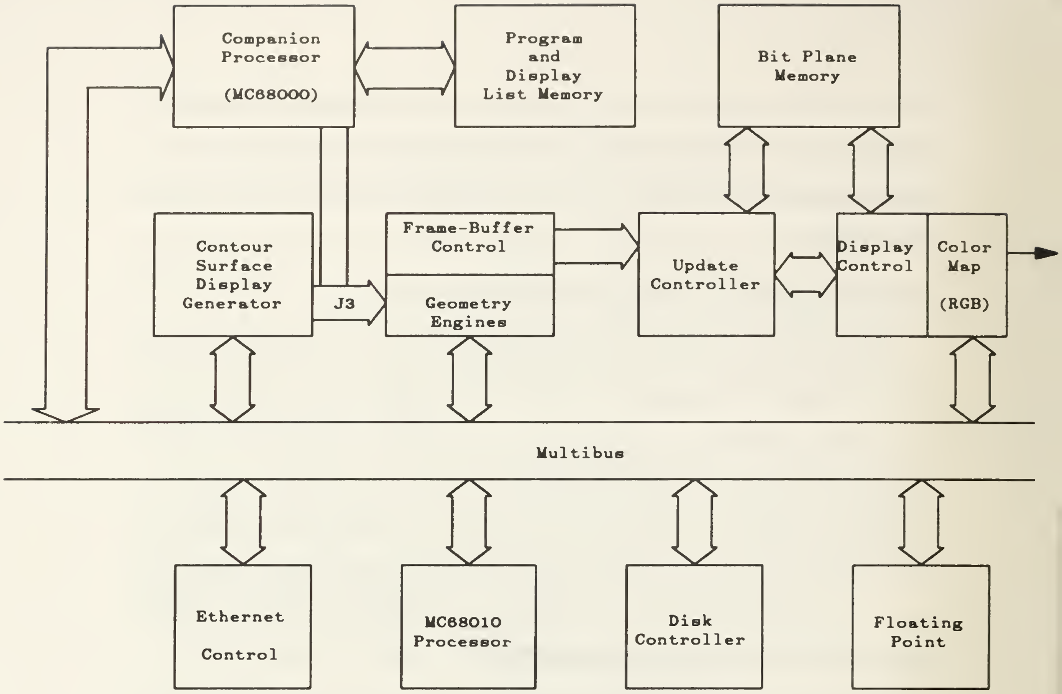


Figure 3
Block Diagram of the Silicon Graphics, Inc. IRIS
(with additional, non-SGI Contour Surface Display Generator)

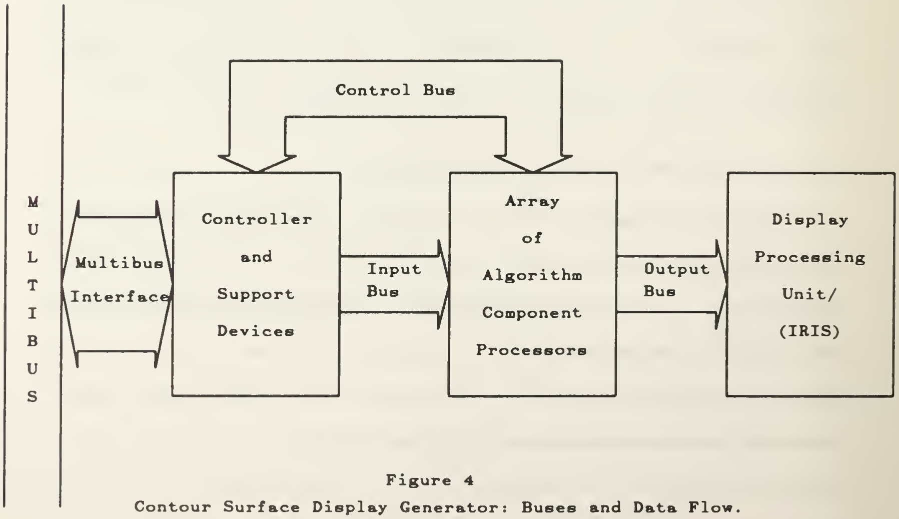

MULTIBUS
Multibus Interface
Control Bus
Controller and Support Devices
Input Bus
Array of Algorithm Component Processors
Output Bus
Display Processing Unit/ (IRIS)

Figure 4
Contour Surface Display Generator: Buses and Data Flow.

Figure 4 depicts the array of algorithm component processors as a single box, with three connections to the outside environment: an input bus for contour levels and subgrids, an output bus for coordinates and drawing instructions, and a bus for controlling the array of processors. A dual bus configuration is chosen to maximize the amount of concurrency in the system due to the autonomous nature of the input with respect to the output.

The input bus is the medium responsible for delivering subgrid definitions and contour levels to the array of algorithm component processors. Because this is the only data required to be transmitted on the bus, the bandwidth of the input bus does not need to be very high. The rate at which subgrid definitions are loaded into the algorithm component processors does not directly affect the real-time capabilities of the system. The real-time capabilities of the contour surface display generator are determined by the rate at which data can be produced in each algorithm component processor. This, in turn, directly affects the rate of output to the display processing unit. The output bus is responsible for delivering the coordinates and drawing instructions to the display processing unit.

The control bus for the contour surface display generator contains all the control lines necessary to manage the data flow on the input side of the system. Two additional control lines are required on the output side of the system to coordinate the two wire handshake between the algorithm component processors and the display processing unit (Geometry Engines).

Control of the array of algorithm component processors involves the integration of several different components. The one which coordinates the operation of all other components is the systems controller (Figure 5). The systems controller converts incoming signals from the Multibus bus master of the Silicon Graphics, Inc. IRIS workstation into signals which make sense to the algorithm component processors. The Multibus bus master is the board in the Multibus Backplane which places the commands on the Multibus. The systems controller is a slave in that it reacts to commands placed on the Multibus.

The component that is responsible for the production of the coordinates and drawing instructions for the contour surface display generator is the algorithm component processor

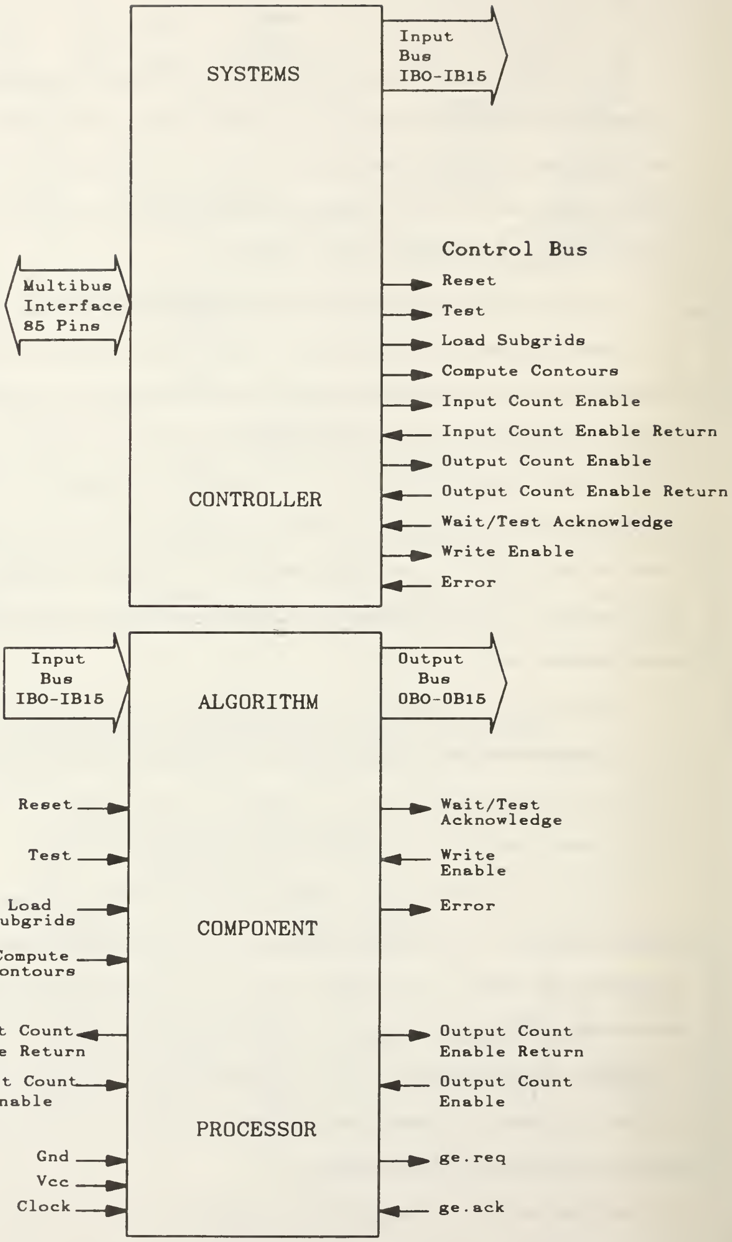


Figure 5
Functional Pin Diagrams of the Systems Controller and the Algorithm Component Processor

(Figure 5). Each of these processors is identical and functions independently in the production of the outputs. We do not go into great detail about that processor other than to state that the processor is a full microprocessor of the Motorola MC68000 class.

The contour surface display generator is connected to the Silicon Graphics, Inc. IRIS graphics system by means of the IEEE standard Multibus Backplane Bus [4]. This Multibus connection provides all inputs to the contour surface display generator. The Multibus interfaces to basically two different classifications of bus modules: (1) Masters - those modules which generate commands, and (2) Slaves - those which respond to commands. The parent processor (MC68000) is the Master module for the graphics system. The contour surface display generator is a slave module in that system.

The output of the contour surface display generator is to the Private Bus of the IRIS system (Figures 3 and 6). The Private Bus is a unidirectional, 16 bit bus dedicated to the provision of coordinate and drawing instructions to the high speed Geometry Engines. Coordination of the transfer of data between the algorithm component processors and the Geometry Engines is done via a two line handshake protocol.

When the contour surface display generator is added to the system, a physical connection to the Geometry Engine pipeline must be shared by both itself and the system processor (the J3 connection of the Geometry Engine board). To enable the user to alternatively route processor and generator data to the Geometry Engines, a hardware switch is added to the system. This hardware provides the system with a way to multiplex the direct path of the Private Bus. A software switch then provides the control of the Private Bus' origin and configuration. When activated, this switch establishes a path from the contour surface display generator. If it is not activated, the IRIS system remains in its original configuration.

4. Hardware Complexity Estimate

The above is a quick overview of the architecture of the contour surface display generator. One of the key components in this system is obviously the algorithm component processor. In [8],

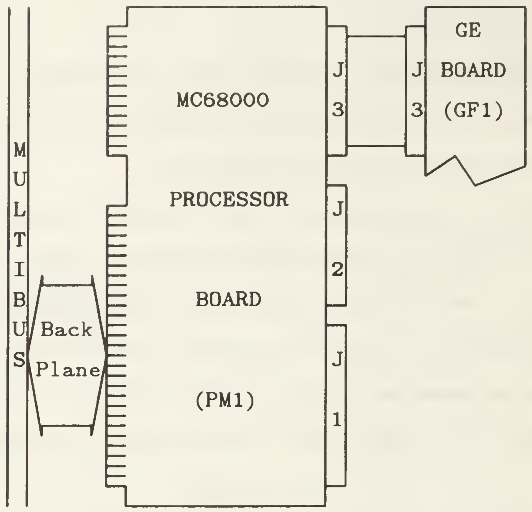


Figure 6a
Silicon Graphics, Inc. IRIS Pipeline Connection
for the Private Bus (Courtesy of Silicon Graphics, Inc.)

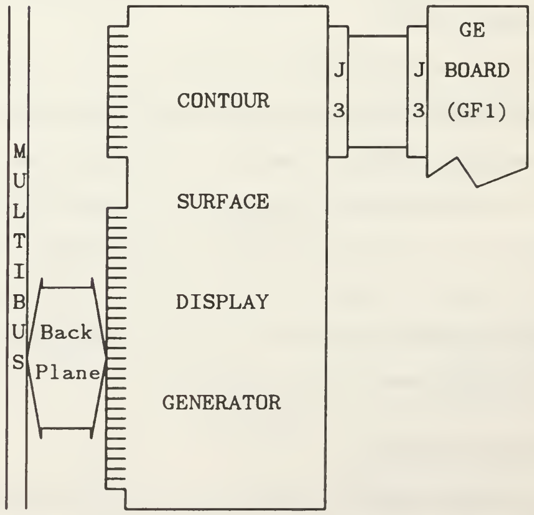


Figure 6b
The J3 Pipeline Connection of the Silicon Graphics, Inc. Private Bus
for the Contour Surface Display Generator

it is determined that 50 algorithm component processors are all that are needed in the system to generate and deliver the average sized picture for a 30 x 30 x 30 grid. In order to determine the feasibility of the complete system, we need a circuit complexity estimate for the size of the algorithm component processor. In [8], we find a summary of the number of transistor equivalent devices necessary. We note only that the total number of devices required for one algorithm component processor is about 660K devices. This number is well below the two million devices per chip level that is currently being produced in research laboratories [5]. For this level of chip complexity, the array of algorithm component processors can be built in less than 25 VLSI chips. At the ten million devices per chip level promised in [7 , this is less than 5 VLSI chips. The design of this system is therefore within the grasp of current technology.

5. Conclusions

This study is intended to give a clearer system description of the elements comprising the contour surface display generator. Some of the deficiencies of earlier research were remedied by defining the control lines of the system. The technological feasibility of the system's implementation is established in two ways. The first is that a realistic number of algorithm component processors is established (50). The second is that the circuit complexity of the algorithm component processor chip is shown to be within the capabilities of current VLSI technology.

We can be assured that once we produce such a system that the applications user will return to us with further hardware demands for either other algorithms, or additional real-time graphics display generation capabilities. In order to respond to these desires for special hardware for select applications graphics algorithms, we need to either justify the special hardware effort on the basis of widespread demand, or make the production of that special hardware inexpensive. We cannot count on the widespread demand for any algorithm for which we desire real-time performance. The only solution then is to make the production of that special hardware inexpensive. The first step in that process is to put together a methodology based upon experience with designing such special purpose display generators. Once that methodology is sufficiently developed, we can then set standards for the production of such systems. We can see an analogy in the world of

VLSI design. The design and production of special VLSI chips came within the possibilities of the university community after standard interfaces were defined for chip production. Hence, we saw the establishment of "silicon foundries". If we extend this idea to that of the production of special hardware for select graphics algorithms of the applications user, this means that somewhere in the future there will be "real-time, graphics foundries". It is towards this direction that we can expect future developments in hardware graphics capabilities to proceed.

6. References

1. Barry, C. D. and Sucher. J. H. "Interactive Real-Time Contouring of Density Maps," American Crystallographic Association Winter Meeting, Honolulu, March 1979, Poster Session.

2. Clark. J. H. and Davis. T. "Workstation Unites Real-Time Graphics with Unix. Ethernet," *Electronics*. October 20, 1983.

3. Faber. D. H., Rutten-Keulemans. E.W.M., and Altona. C. "Computer Plotting of Contour Maps: An Improved Method," *Computers & Chemistry*, Vol. 3, pp. 51-55, Great Britain: Pergamon Press Ltd., 1979.

4. Intel Corporation Technical Publication, *Intel Multibus Specification*, Intel Corporation, Santa Clara. California, 1982.

5. Micronews "IBM Experimental Million Bit Memory Chip." *IEEE Micro*, Vol. 4, No. 4 (August 1984). p. 119.

6. Newman. William H.. and Sproull, Robert F. *Principles of Interactive Graphics*, Second Edition. New York: McGraw-Hill, 1979.

7. Uhr, Leonard *Algorithm-Structured Computer Arrays and Networks*, Orlando, Florida: Academic Press. 1984.

8. Walker. Robert A. and Zyda, Michael J "An Integrated Systems Architecture for Real-Time Contour Surface Display Generation." Technical Report NPS52-85-010. Monterey, California: Department of Computer Science, Naval Postgraduate School, August 1985.

9. Wright, T. and Humbrecht, J. "ISOSRF -- An Algorithm for Plotting Iso-Valued Surfaces of a Function of Three Variables," *Computer Graphics: A Quarterly Report of SIGGRAPH-ACM*. Vol. 13. No. 2 (August 1979), pp. 182-189.

10. Zyda. Michael J. "The Feasibility of a Multiprocessor Architecture for Real-Time Contour Surface Display Generation." Technical Report NPS52-84-025, Monterey. California: Department of Computer Science, Naval Postgraduate School, December 1984.

11. Zyda. Michael J. "Real-Time Contour Surface Display Generation," Technical Report NPS52-84-013, Monterey, California: Department of Computer Science, Naval Postgraduate School, September 1984.

12. Zyda, Michael J. "A Decomposable Algorithm for Contour Surface Display Generation," Technical Report NPS52-84-011, Monterey, California: Department of Computer Science, Naval Postgraduate School, August 1984.

13. Zyda, Michael J. *Algorithm Directed Architectures for Real-Time Surface Display Generation,* D.Sc. Dissertation, Dept. of Computer Science, Washington University, St. Louis, Missouri, January 1984.

14. Zyda, Michael J. "A Contour Display Generation Algorithm for VLSI Implementation," *Selected Reprints on VLSI Technologies and Computer Graphics,* Compiled by Henry Fuchs, p. 459, Silver Spring, Maryland: IEEE Computer Society Press, 1983.

15. Zyda, Michael J. "A Contour Display Generation Algorithm for VLSI Implementation," *Computer Graphics: A Quarterly Report of SIGGRAPH-ACM,* Vol. 16, No. 3 (July 1982), p. 135.

16. Zyda, Michael J. "Multiprocessor Considerations in the Design of a Real-Time Contour Display Generator," Technical Memorandum 42, St. Louis: Department of Computer Science, Washington University, December 1981.

17. Zyda, Michael J. "Joystick Driven Display Rotation and Control Console Management," Technical Memorandum 24, St. Louis: Department of Computer Science, Washington University, November 1980.

Distribution List for Papers Written by Michael J. Zyda

Defense Technical Information Center, Cameron Station, Alexandria, VA 22314	2 copies
Library, Code 0142 Naval Postgraduate School, Monterey, CA 93943	2 copies

Center for Naval Analyses,
2000 N. Beauregard Street,
Alexandria, VA 22311

Director of Research Administration.
Code 012,
Naval Postgraduate School.
Monterey, CA 93943

Dr. Henry Fuchs.
208 New West Hall (035A),
University of North Carolina,
Chapel Hill, NC 27514

Dr. Kent R. Wilson.
University of California, San Diego
B-014,
Dept. of Chemistry,
La Jolla, CA 92093

Dr. Guy L. Tribble, III
900 Waverly St.
Palo Alto, California 94301

Bill Atkinson.
Apple Computer,
20525 Mariani Ave,
Cupertino, CA 95014

Dr. Victor Lesser,
University of Massachusetts, Amherst
Dept. of Computer and Information Science,
Amherst, MA 01003

Dr. Gunther Schrack.
Dept. of Electrical Engineering,
University of British Columbia.
Vancouver, B.C., Canada V6T 1W5

Dr. R. Daniel Bergeron,
Dept. of Computer Science.
University of New Hampshire.
Durham. NH 03824

Dr. Ed Wegman,
Division Head,
Mathematical Sciences Division,
Office of Naval Research.
800 N. Quincy Street,
Arlington, VA 22217-5000

Dr. Gregory B. Smith,
ATT Information Systems,
190 River Road,
Summit, NJ 07901

Dr. Lynn Conway,
University of Michigan,
263 Chrysler Center,
Ann Arbor, MI 48109

Dr. John Lowrance.
SRI International,
333 Ravenswood Ave.
Menlo Park, CA 94025

Dr. David Mizell,
Office of Naval Research.
1030 E. Green St.
Pasadena, CA 91106

Dr. Richard Lau,
Office of Naval Research.
Code 411,
800 N. Quincy St.
Arlington, VA 22217-5000

Dr. Y.S. Wu,
Naval Research Laboratory,
Code 7007,
Washington, D.C. 20375

Dr. Joel Trimble,
Office of Naval Research,
Code 251,
Arlington, VA 22217-5000

Robert A. Ellis,
Calma Company,
R & D Engineering,
525 Sycamore Dr., M/S C51O
Milpitas, CA 95035-7489

Dr. James H. Clark,
Silicon Graphics, Inc.
2011 Stierlin Road,
Mountain View, CA 94043

Edward R. McCracken,
Silicon Graphics, Inc.
2011 Stierlin Road,
Mountain View, CA 94043

Shinji Tomita,
Dept. of Information Science,
Kyoto University,
Sakyo-ku, Kyoto, 606, Japan

Hiroshi Hagiwara,
Dept. of Information Science,
Kyoto University,
Sakyo-ku, Kyoto, 606, Japan

Dr. Alain Fournier,
Dept. of Computer Science,
University of Toronto,
Toronto, Ontario, Canada
M5S 1A4

Dr. Andries Van Dam,
Dept. of Computer Science,
Brown University,
Providence, RI 02912

Dr. Brian A. Barsky,
Berkeley Computer Graphics Laboratory,
Computer Sciences Division,
Dept. of Electrical Engineering and Computer Sciences,
University of California,
Berkeley, CA 94720

Dr. Ivan E. Sutherland,
Carnegie Mellon University,
Pittsburg, PA 15213

Dr. Turner Whitted,
New West Hall (035A),
University of North Carolina,
Chapel Hill, NC 27514

Dr. Robert B. Grafton,
Office of Naval Research,
Code 433,
Arlington, Virginia 22217-5000

Professor Eihachiro Nakamae,
Electric Machinery Laboratory,
Hiroshima University,
Higashihiroshima 724, Japan

Carl Machover,
Machover Associates,
199 Main Street,
White Plains, New York 10601

Dr. Buddy Dean,
Naval Postgraduate School,
Code 52, Dept. of Computer Science,
Monterey, California 93943

Earl Billingsley,
43 Fort Hill Terrace,
Northhampton, MA 01060

Dr. Jan Cuny,
University of Massachusetts, Amherst
Dept. of Computer and Information Science,
Amherst, MA 01003

Robert Lum,
Silicon Graphics, Inc.
2011 Stierlin Road,
Mountain View, CA 94043

Jeff Hausch,
Silicon Graphics, Inc.
2011 Stierlin Road,
Mountain View, CA 94043

Lt. Robert A. Walker,
Naval Sea Systems Command (SEA 61YM),
Department of the Navy,
Washington, DC 20362-5101

Dr. Barry L. Kalman,
Washington University,
Department of Computer Science,
Box 1045,
St. Louis, Missouri 63130

Dr. Wm. Randolph Franklin,
Electrical, Computer, and Systems Engineering Department,
Rensselaer Polytechnic Institute,
Troy, New York 12180-3590

Dr. Gershon Kedem,
Microelectronics Center of North Carolina,
PO Box 12889,
3021 Cornwallis Road,
Research Triangle Park,
North Carolina 27709

Dr. Branko J. Gerovac,
Digital Equipment Corporation,
150 Locke Drive LMO4/H4, Box 1015
Marlboro, Massachusetts 01752-9115

Robert A. Schumacker,
Evans and Sutherland,
PO Box 8700,
580 Arapeen Drive,
Salt Lake City, Utah 84108

R. A. Dammkoehler,
Washington University,
Department of Computer Science,
Box 1045,
St. Louis, Missouri 63130

Dr. Lynn Ten Eyck,
Interface Software,
79521 Highway 99N,
Cottage Grove, Oregon 97424

Toshiaki Yoshinaga,
Hitachi Works, Hitachi Ltd.
1-1, Saiwaicho 3 Chome,
Hitachi-shi, Ibaraki-ken,
317 Japan

Takatoshi Kodaira,
Omika Works, Hitachi Ltd.
2-1, Omika-cho 5-chome,
Hitachi-shi, Ibaraki-ken,
319-12 Japan

Atsushi Suzuki,
Hitachi Engineering, Co. Ltd.
2-1, Saiwai-cho 3-Chome,
Hitachi-shi, Ibaraki-ken,
317 Japan

Toshiro Nishimura,
Hitachi Engineering, Co. Ltd.
2-1, Saiwai-cho 3-Chome,
Hitachi-shi, Ibaraki-ken,
317 Japan

Dr. John Staudhammer,
Dept. of Electrical Engineering,
University of Florida,
Gainesville, Florida 32611

Dr. Lewis E. Hitchner,
Computer and Information Science Dept.
237 Applied Science Building,
University of California at Santa Cruz,
Santa Cruz, California 95064

Dr. Pat Mantey,
Computer Engineering Department,
University of California at Santa Cruz,
Santa Cruz, California 95064

Dr. Walter A. Burkhardt,
University of California, San Diego
Dept. of Computer Science.
La Jolla, California 92093

P. K. Rustagi.
Silicon Graphics. Inc.
2011 Stierlin Road.
Mountain View. CA 94043

Peter Broadwell.
Silicon Graphics, Inc.
2011 Stierlin Road.
Mountain View. CA 94043

Norm Miller.
Silicon Graphics, Inc.
2011 Stierlin Road.
Mountain View, CA 94043

Dr. Tosiyasu L. Kunii.
Department of Information Science.
Faculty of Science.
The University of Tokyo.
7-3-1 Hongo. Bunkyo-ku. Tokyo 113,
Japan

Dr. Kazuhiro Fuchi.
Institute for New Generation Computer Technology.
Mita-Kokusai Building 21FL.
1-4-28 Mita. Minato-ku. Tokyo 108. Japan

Tony Loeb.
Silicon Graphics. Inc.
1901 Avenue of the Stars,
Suite 1774.
Los Angeles. CA 90067

Kevin Hammons.
NASA AMES-Dryden Flight Research Facility,
PO Box 273.
Mail Stop OFI.
Edwards, California 93523

Sherman Gee,
Code 221,
Office of Naval Technology,
800 N. Quincy St.
Arlington, VA 22217

Dr. J.A. Adams,
Department of Mechanical Engineering,
US Naval Academy,
Annapolis, MD 21402

Dr. David F. Rogers,
Dept. of Aerospace Engineering,
US Naval Academy,
Annapolis, MD 21402

Dr. Robert F. Franklin,
Environmental Research Institute of Michigan,
PO Box 8618,
Ann Arbor, MI 48107

LT Mark W. Hartong,
900 Cambridge Dr 17,
Benicia, CA 94510

Capt. Mike Gaddis,
DCA/JDSSC/C720,
1860 Wiehle Ave
Reston, VA 22090

Lt. Cdr. Patrick G. Hogan, USN
102 Borden Avenue,
Wilmington, North Carolina 28403

Dr. Edwin Catmull,
LucasFilm,
PO Box 2009,
San Rafael, CA 94912

Dr. John Beatty,
Computer Science Department,
University of Waterloo,
Waterloo, Ontario,
Canada N2L 3G1

Dr. James Foley,
George Washington University,
Dept. of Electrical Engineering and Computer Science,
Washington, D.C. 20052

Dr. Donald Greenberg,
Cornell University,
Program of Computer Graphics,
Ithaca, NY 14853

Dr. Leo J. Guibas,
Systems Research Center,
Digital Equipment Corporation,
130 Lytton Avenue,
Palo Alto. CA 94301

Dr. S. Ganapathy,
Ultrasonic Imaging Laboratory,
Dept. of Electrical and Computer Engineering.
University of Michigan.
Ann Arbor. MI 48109

Dr. Hank Christiansen.
Brigham Young University.
Dept. of Civil Engineering.
368 Clyde Bldg.
Provo. Utah 84602

Dr. Thomas A. DeFanti.
Dept. of Electrical Engineering & Computer Science.
University of Illinois at Chicago.
Box 4348.
Chicago. IL 60680

Dr. Lansing Hatfield.
Lawrence Livermore National Laboratory,
7000 East Avenue.
PO Box 5504. L-156,
Livermore. CA 94550